1、选择器

1. 结构伪类

特点：先匹配再排序

- :first-of-type 选择某个元素的第一个元素
- :last-of-type 选择某个元素的最后一个元素
- :nth-of-type() 选择父元素中特定的某种类型的子元素

数字 1 2 3 4

公式 2n 2n-1

关键字 even odd

- :nth-of-last-type 选择特定类型元素的最后一个元素

```
<style>
        ul li:first-of-type {
            background-color: pink;
        }

        ul li:last-of-type {
            background-color: pink;
        }

        ul li:nth-of-type(even) {
            background-color: skyblue;
        }

        /* nth-child 会把所有的盒子都排列序号 */
        /* 执行的时候首先看  :nth-child(1) 之后回去看 前面 div */

        section div:nth-child(1) {
            background-color: red;
        }

        /* nth-of-type 会把指定元素的盒子排列序号 */
        /* 执行的时候首先看  div指定的元素  之后回去看 :nth-of-type(1) 第几个孩子 */
        section div:nth-of-type(1) {
            background-color: blue;
        }
    </style>
```

```
    <body>
    <ul>
        <li>我是第1个孩子</li>
        <li>我是第2个孩子</li>
        <li>我是第3个孩子</li>
        <li>我是第4个孩子</li>
        <li>我是第5个孩子</li>
        <li>我是第6个孩子</li>
        <li>我是第7个孩子</li>
        <li>我是第8个孩子</li>
    </ul>
    <!-- 区别 -->
    <section>
        <p>光头强</p>
        <div>熊大</div>
        <div>熊二</div>
    </section>
</body>
```

2. 特殊伪类

- :only-child 表示父元素下没有兄弟元素，只有一个元素
- :only-of-type 表示父元素下没有相同的兄弟元素，可以有相同类型的元素
- :empty 表示选择空节点(里面没有任何元素)
- :not 表示排除某个元素

写法：div:not(p) {

属性："属性值"

}

- :root 匹配根节点（不在有父元素）

```
p:only-of-type {
    color: red;
 }
div:empty {
    width: 100px;
    height: 100px;
    background-color: red;
 }
.demo:not(p) {
    color: red;
```

```
}
:root {
    background-color: pink;
}
```

3. 目标伪类

- :target

id值被当成链接就被称为目标

4. 交集选择器

- 表示既什么又什么，连写

格式:

选择器选择器 {

属性:"属性值"

}

5. 并集选择器

- 一般作用公共样式，用逗号隔开

格式:

选择器,选择器 {

属性:"属性值"

}

6. 后代选择器

- 会选择前一个选择器的所有后代，用空格隔开

格式:

选择器 选择器 {

属性:"属性值"

}

7. 子代选择器

- 只会选择所有相同类型的子代

格式:

选择器>选择器 {

属性:"属性值"

}

8. 兄弟选择器

- +：选择下一个相邻的兄弟

选择器+选择器 {
属性:"属性值"
}

- ~：选择下面所用相同类型的兄弟

选择器~选择器 {
属性:"属性值"
}

9. 属性选择器

格式:
[属性] {
属性:"属性值"
}
或者
[属性="属性值"] {
属性:"属性值"
}

- [属性^="属性值"] 表示以某属性值为开头
- [属性$="属性值"] 表示以某属性值为结尾
- [属性*="属性值"] 表示包含某属性值

```
<style>
        /* 必须是input 但是同时具有 value这个属性 选择这个元素  [] */
        /* input[value] {
            color:pink;
        } */
        /* 只选择 type =text 文本框的input 选取出来 */
        input[type=text] {
            color: pink;
        }

        /* 选择首先是div 然后 具有class属性 并且属性值 必须是 icon开头的这些元素 */
        div[class^=icon] {
            color: red;
        }

```

```
        section[class$=data] {
            color: blue;
        }

        div.icon1 {
            color: skyblue;
        }

        /* 类选择器和属性选择器 伪类选择器 权重都是 10 */
    </style>
```

10. 伪元素选择器

- ::first-line 选择浏览器第一行
- ::first-letter 选择浏览器第一个字符
- ::selection 表示鼠标选中的样式
- ::before 在元素内部的结束位置创建一个元素 要配合content属性使用
- ::after 在元素内部的开始位置创建一个元素 要配合content属性使用

```
<style>
        div {
            width: 200px;
            height: 200px;
            background-color: pink;
        }

        /* div::before 权重是2 */
        div::before {
            /* 这个content是必须要写的 */
            /* display: inline-block; */
            content: '我';
            /* width: 30px;
            height: 40px;
            background-color: purple; */
        }

        div::after {
            content: 'NI';
        }
</style>
```

一个"."表示类选择器，":"伪类选择器，"::"表示伪元素选择器

2、CSS三大特性

1.层叠性

- 相同选择器设置相同的样式，此时一个样式就会覆盖另一个冲突的样式。
- 层叠性主要解决样式冲突的问题
- 层叠性原则：样式冲突，遵循的原则是就近原则

3、继承性

- CSS中的继承：子标签会继承父标签里面的某些样式
- 如文本颜色、字号、行高等

4、优先级

- 当同一个元素指定多个选择器，就会有优先级的产生

```
<head>
<style>
      div{
         color:pink;
      }
      .text {
      color:red;
      }
</style>
</head>
<body>
       <div class="text">好看</div>
</body>
0111213
```

- 选择器相同，则执行层叠性
- 选择器不同，则按权重

选择器	权重
继承或者 *	0，0，0，0
元素选择器（标签选择器）	0，0，0，1
类选择器，伪类选择器	0，0，1，0
ID选择器	0，1，0，0

行内样式style=""	1，0，0，0
！important重要的	无穷大

- 类选择器权重为10
- 伪类选择器权重为10
- ID选择器权重为100

优先级注意点

- 等级判断是从左到右，如果某一位数值相同，则判断下一位数值
- 继承的权重是0
- 权重可以叠加，但是永远不会有进位

权重的叠加

- 权重叠加：如果是复合选择器，则会有权重的叠加，需要计算权重
- div ul li----------> 0,0,0,3
- .nav ul li ---------->0,0,1,2
- a:hover ---------->0,0,1,1
- .nav a---------->0,0,1,1

5、display显示

- 文档流：

从上往下排列，有独占一行的就一行显示，不能一行显示就从左往右依次排列

- display:block; 块级元素 h1-h6 p div ul li ol

特点：

独占一行

设置宽高生效

- display:inline; 行内元素 span i em s del b strong u ins

特点：

不独占一行

设置宽高不生效

- display:inline-block; 行内块级元素 input img

特点:

不独占一行

设置宽高生效

会以文字基线进行对齐，里面没有元素则底部就是文字基线

- display:none 元素消失 脱离文档流 原来的位置不在了
- visibility:hidden 元素消失 原来的位置依然存在

11. 元素的显示与隐藏

- 当点击第一个盒子时，第二个盒子呈现显示或者隐藏
- 可以使用：hover伪元素选择器，在使用兄弟选择器

```
.demo:hover+.demo2 {
      display: block;
  }
```

6、盒子模型

布局的本质就是盒子

盒模型由外到内由以下四个部分组成

标准盒子模型：content(height+width)+padding+border+margin

1. margin 外边距

- 取值顺序：上右下左

margin:10px; 上右下左都是10px

margin-left: 左外边距

margin-top: 上外边距

margin-right: 右外边距

margin-bottom: 下外边距

margin: 10px 20px; 上下10px 左右20px

margin: 10px 20px 30px; 上10px 左右20px 下30px

margin: 10px 20px 30px 40px;

margin:0 auto; （上下 左右） 盒子居中

margin: auto; 左右自适应达到水平居中的效果

- 使用条件：

1.自身得有宽度

2.必须是块级元素

2. border 边框

- 边框的宽度： 以px为单位的值
- 边框的样式：

solid： 实线

dashed： 虚线

dotted：点线

double: 双实线

- 边框的颜色

单词 rgb hex

border-top: 3px solid slateblue; 上边框

border-bottom: 3px solid slateblue; 下边框

border-left: 3px solid slateblue; 左边框

border-right: 3px solid slateblue; 右边框

- 复合写法形式：

border: 1px solid red; 四个方向

- 边框圆角：border-radius

取值顺序：左上 右上 右下 左下

border-radius: 30px; 四个角

border-top-left-radius 左上

border-top-right-radius 右上

border-bottom-left-radius 左下

border-bottom-right-radius 右下

3. padding 内边距 border 和 内容之间的间隙

取值顺序：上右下左

padding:10px; 上右下左都是10px

padding-left: 左内边距

padding-top: 上内边距

padding-right: 右内边距

padding-bottom: 下内边距

padding: 10px 20px; 上下10px 左右20px

padding: 10px 20px 30px; 上10px 左右20px 下30px

padding: 10px 20px 30px 40px;

4. content： (height+width)

5. 怪异盒子模型：(content+padding+border)(height+width)+margin

6. 标准盒子转怪异盒子: box-sizing:border-box;

(自身要有宽度或高度)

7、CSS三角形

```
.jd span {
      width: 0;
      height: 0;
      /* 为了照顾兼容性 */
       line-height: 0;
       font-size: 0;

       border: 5px solid transparent;
       border-bottom-color: blue;
    }
```

8、margin存在的问题

1. 外边距重合

解决办法：

- 给父元素使用padding
- 给父元素增加一个边框border: 0.01px solid transparent;
- 给父元素增加一个overflow:hidden(超出部分隐藏) 推荐！！！

overflow:hidden 触发了BFC机制，让父元素形成一个独立的容器，里面的元素不会影响外面元素的排列方式

2. 外边距嵌套

解决办法：

- 避免问题 !!!
- 给某一个元素增加父级，给父级设置overflow:hidden

```
<style>
        /* 清除内外边距
        *{
            padding: 0;
            margin: 0;
        } */
        .father {
            width: 300px;
            height: 200px;
            overflow: hidden;
            /* margin-top: 20px;
            盒子嵌套设置外边距会塌陷 */
            background-color: pink;
            /* margin: 0 auto;  居中对齐 */
        }

        /* 盒子嵌套设置外边距会塌陷 解决办法：给父元素设置边框；给父元素设置内边距；给父元素加overflow:hidden */
        .son {
            width: 100px;
            height: 100px;
            margin-top: 20px;
            /* 盒子嵌套设置外边距会塌陷 */
            background-color: blue;
        }
```

```
    </style>
```

3. 页面居中

- 父元素设置宽度，外边距marjin:auto; 用子元素撑开父元素

```
.demo {
        width: 1226px;
        margin: auto;
    }
.son {
        height: 200px;
        width: 200px;
        background-color: pink;
    }
```